Facultad de ingeniería de la UBA

66.20 Organización de Computadoras

Trabajo práctico N°1
Programación MIPS

Segundo cuatrimestre de 2020

Integrante	Padrón	Correo electrónico
Calvo, Mateo Iván	98290	macalvo@fi.uba.ar
Jamilis Netanel David	99093	njamilis@fi.uba.ar
Sabao Tomás	99437	tsabao@fi.uba.ar

Fecha de entrega: 10/11/2020
Grupo 10

Índice

1. Documentación

Aquí se discuten los aspectos de documentación e implementación que se consideraron más interesantes.

1.1. Stackframe de la función string_hash_more

La función implementada en *assembly*, `string_hash_more`, tiene un *stackframe* compuesto solamente por la *Saved Register Area*. Lo anterior se da como resultado de ser una función *non-leaf*, que no necesita dejar espacio en el stackframe para el *Argument Building Area* (ABA) además que se decidió realizar las operaciones intermedias usando únicamente registros del procesador. Esta es la razón por la cual la *Local and Temporary Area* (LTA) se encuentra vacía.

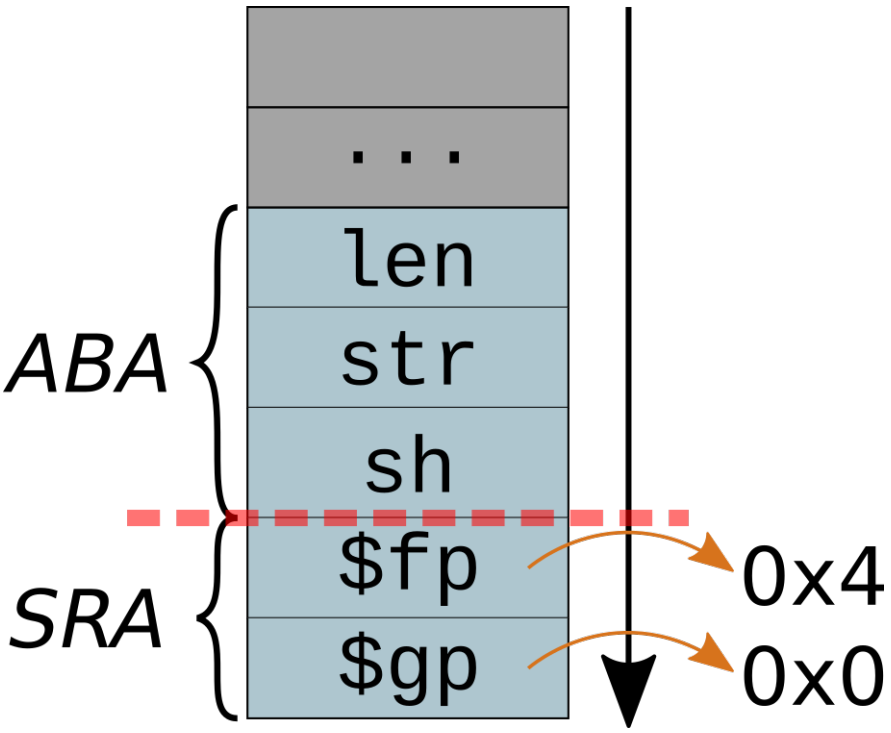

Figura 1: Porción del *Stack*, en la cual se puede observar el *Stackframe* de la función `string_hash_more`, compuesto únicamente por la *SRA* (direcciones relativas 0 y 4). En las direcciones superiores, puede observarse el *ABA* de la función *caller*.

2. Compilación y ejecución

A continuación se muestra la manera de compilar y ejecutar el programa. El archivo `Makefile` necesario se encuentra anexado a esta entrega digital. En el segundo ejemplo de ejecución se observan los modos de ejecución del programa, y se indica cómo pasar parámetros cuando estos sean requeridos.

Compilación del ejecutable

```
root@debian-stretch-mips:/tp1/src# make tp1
cc -Wall -Werror -pedantic -o tp1 main.c hash.c hash.S
hasheador.c
```

Ejecución

```
root@debian-stretch-mips:/tp1/src# ./tp1 -h
Usage:
tp1 -h
tp1 -V
tp1 -i in_file -o out_file
Options:
-V, --version
Print version and quit.
-h, --help
Print this information and quit.
-i, --input
Specify input stream/file, "for stdin.
-o, --output
Specify output stream/file, "for stdout.
Examples:
tp1 <in.txt >out.txt
cat in.txt | tp1 -i - >out.txt
```

Ejecución

```
root@debian-stretch-mips:/tp1/src# ./tp1 -V
Version 0.1
root@debian-stretch-mips:/tp1/src#
echo 1 | ./tp1 -o -
0x91ff4b5b 1
```

3. Pruebas

Para realizar las pruebas que verificaran el correcto funcionamiento del código implementado, se utilizaron las propuestas por la cátedra. No se ahondó en un desarrollo más abarcativo de pruebas debido a la poca extensión de la función a implementar, las principales dificultades se encontraron en el lenguaje ensamblador. Sin embargo, se realizaron varias sesiones de *debugging* mediante el programa `gdb`, que permitieron encontrar y corregir errores al poder observar el estado de los registros entre instrucción e instrucción.

3.1. Regresiones

Para compilar y ejecutar las regresiones, debe hacerse:

Compilación y ejecución de regresiones

```
root@debian-stretch-mips:/tp1/src# make test
cc -Wall -g -o regressions regressions.c hash.c hash.S
./regressions
0xcc2b6c5a 66.20 Organizacion de Computadoras
0xcb5af1f1 TP 1 - Segundo Cuatrimestre, 2020
0xcb5af1f1
0xd788c5a5 Archivo de prueba TP 1.
0x91ff4b5b 1
```

Un correcto funcionamiento provocará que se muestren las líneas anteriores en consola, en otro caso se producirá un error.

4. Código fuente y MIPS32

Tanto el código fuente como el MIPS32 se encuentran en la sección de apéndices. En el apéndice A se encuentra el código fuente y MIPS, mientras que en el apéndice B se puede encontrar el código MIPS32.

Apéndices

A. Código fuente C y MIPS

A continuación se encuentra el código fuente del programa, excluyendo al archivo de regresiones.

```
 1: #ifndef _HASH_H_INCLUDED_
 2: #define _HASH_H_INCLUDED_
 3:
 4: #define STRING_HASH_INIT 1
 5: #define STRING_HASH_MORE 2
 6: #define STRING_HASH_DONE 3
 7:
 8: #ifndef __ASSEMBLER__
 9:
10: #include <string.h>
11: #include <assert.h>
12:
13: typedef struct {
14:         int8_t flag;
15:         int32_t hash;
16:         size_t size;
17: } string_hash;
18:
19:
20: extern void string_hash_init(string_hash *);
21: extern void string_hash_more(string_hash *, char *, size_t);
22: extern void string_hash_done(string_hash *);
23: extern int32_t string_hash_value(string_hash *);
24: #endif
25:
26: #endif
```

```
 1: #include <stdint.h>
 2: #include <assert.h>
 3:
 4: #include "hash.h"
 5:
 6: void
 7: string_hash_init(string_hash *h)
 8: {
 9:         h->flag = STRING_HASH_INIT;
10:         h->hash = 0;
11:         h->size = 0;
12: }
13:
14: void
15: string_hash_done(string_hash *sh)
16: {
17:         assert(sh->flag == STRING_HASH_INIT || sh->flag == STRING_HASH_MORE);
18:
19:         if ((sh->hash ^= sh->size) == -1)
20:                 sh->hash = -2;
21:
22:         sh->flag = STRING_HASH_DONE;
23: }
24:
25: int32_t
26: string_hash_value(string_hash *sh)
27: {
28:         return sh->hash;
29: }
```

```c
#ifndef HASHEADOR_H
#define HASHEADOR_H

#include <stdint.h>
#include <stdio.h>

#define MENSAJE_USO "\
Usage:\n\
tp1 -h\n\
tp1 -V\n\
tp1 -i in_file -o out_file\n\
Options:\n\
-V, --version\n\
Print version and quit.\n\
-h, --help\n\
Print this information and quit.\n\
-i, --input\n\
Specify input stream/file, \"-\" for stdin.\n\
-o, --output\n\
Specify output stream/file, \"-\" for stdout.\n\
Examples:\n\
tp1 < in.txt > out.txt\n\
cat in.txt | tp1 -i - > out.txt\n"

#define MENSAJE_VERSION "version 0.1\n"

#define ESTADO_OK 0
#define ESTADO_ERROR -1
#define PARAM_AYUDA "-h"
#define PARAM_VERSION "-V"
#define PARAM_INPUT "-i"
#define PARAM_OUTPUT "-o"

#define MODO_AYUDA 0
#define MODO_VERSION 1
#define MODO_CORRER 2

#define RESULTADO_OK 0
#define RESULTADO_ERROR -1

typedef struct hasheador {
    FILE* entrada;
    FILE* salida;
    int estado;
    int modo;
} hasheador_t;

/*
Inicializa un hasheador, con sus parametros recibidos.
Params:
    hasheador: un puntero valido
    n_parametros: cantidad de parametros recibidos
    parametros: array de strings que contiene los parametros
Retorno:
    ESTADO_OK (0) si no hay fallos, ESTADO_ERROR (-1) en caso contrario
*/
int hasheador_inicializar(hasheador_t* hasheador, int n_parametros, \
    const char* parametros[]);

/*
Destruye y libera todos los recursos adquiridos en hasheador.
*/
void hasheador_destruir(hasheador_t* hasheador);

/*
Ejecuta el hasheador segun el modo elegido en los parametros recibidos
*/
int hasheador_correr(hasheador_t* hasheador);
```

```
69:
70: /*
71: Para cada linea de la entrada del hasheador, calcula su valor de hash, e
72: imprime el resultado seguido de la linea.
73: */
74: int hasheador_hashear_archivo(hasheador_t* hasheador);
75:
76: #endif
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

#include "hasheador.h"
#include "hash.h"


/*
Devuelve true si el null-terminated string str se encuentra en array.
Si se especifica pos, se devuelve alli la posicion.
Params:
    str: string que se quiere buscar
    array: arreglo de strings
    array_size: tamanio de array
    pos: puntero donde se devuelve la posicion, NULL para ignorar
        comportamiento
Retorno:
    true si str se encuentra en array, false en caso contrario.
    si str se encuentra en array y pos no es NULL, pos contiene
    la posicion de str en array.
    Si str no se encuentra en array, pos no se altera.
*/
static bool str_en_array(const char* str, \
    const char** array, size_t array_size, size_t* pos) {

        for (size_t i = 0; i < array_size; i++) {
            if (!strcmp(array[i], str)) {
                if (pos) {
                    *pos = i;
                }
                return true;
            }
        }
        return false;
}

int hasheador_inicializar(hasheador_t* hasheador, int n_parametros, \
    const char* parametros[]) {

    hasheador->modo = MODO_CORRER;
    hasheador->estado = ESTADO_OK;

    if (str_en_array(PARAM_AYUDA, parametros, n_parametros, \
        NULL)) {
        hasheador->modo = MODO_AYUDA;
    } else if (str_en_array(PARAM_VERSION, parametros, \
        n_parametros, NULL)) {
        hasheador->modo = MODO_VERSION;
    }

    size_t posicion = 0;

    if (str_en_array(PARAM_INPUT, parametros, n_parametros, \
        &posicion)) {
        if (*parametros[posicion + 1] == '-') {
            hasheador->entrada = stdin;
        } else {
            hasheador->entrada = fopen(parametros[posicion + 1], "r");
            if (!hasheador->entrada) {
                hasheador->estado = ESTADO_ERROR;
                goto fin;
            }
        }
    } else {
        hasheador->entrada = stdin;
    }
```

```
 69:
 70:     if (str_en_array(PARAM_OUTPUT, parametros, n_parametros, \
 71:         &posicion)) {
 72:         if (*parametros[posicion + 1] == '-') {
 73:             hasheador->salida = stdout;
 74:         } else {
 75:             hasheador->salida = fopen(parametros[posicion + 1], "w");
 76:             if (!hasheador->salida) {
 77:                 hasheador->estado = ESTADO_ERROR;
 78:                 if (hasheador->entrada != stdin) {
 79:                     fclose(hasheador->entrada);
 80:                 }
 81:             }
 82:         }
 83:     } else {
 84:         hasheador->salida = stdout;
 85:     }
 86:     fin:
 87:     return hasheador->estado;
 88: }
 89:
 90: void hasheador_destruir(hasheador_t* hasheador) {
 91:     if (hasheador->estado == ESTADO_ERROR) {
 92:         return;
 93:     }
 94:     if (hasheador->entrada != stdin) {
 95:         fclose(stdin);
 96:     }
 97:     if (hasheador->salida != stdout) {
 98:         fclose(stdout);
 99:     }
100: }
101:
102: int hasheador_correr(hasheador_t* hasheador) {
103:     int resultado = 0;
104:     if (hasheador->estado == ESTADO_ERROR) {
105:         printf("%s", MENSAJE_USO);
106:         resultado =  RESULTADO_ERROR;
107:     } else if (hasheador->modo == MODO_AYUDA) {
108:         printf("%s", MENSAJE_USO);
109:         resultado = RESULTADO_OK;
110:     } else if (hasheador->modo == MODO_VERSION) {
111:         printf("%s", MENSAJE_VERSION);
112:         resultado = RESULTADO_OK;
113:     } else { // Modo correr
114:         resultado = hasheador_hashear_archivo(hasheador);
115:     }
116:     return resultado;
117: }
118:
119: int hasheador_hashear_archivo(hasheador_t* hasheador) {
120:
121:     string_hash sh;
122:
123:     char* line_ptr = NULL;
124:     size_t tamanio = 0;
125:     int resultado = 0;
126:     while ((resultado = getline(&line_ptr, &tamanio, \
127:         hasheador->entrada)) != -1) {
128:
129:         string_hash_init(&sh);
130:         string_hash_more(&sh, line_ptr, tamanio);
131:         string_hash_done(&sh);
132:         fprintf(hasheador->salida, "0x%08x %s", \
133:             string_hash_value(&sh), line_ptr);
134:     }
135:     if (feof(hasheador->entrada)) {
136:         resultado = RESULTADO_OK;
```

```
137:     } else {
138:         resultado = RESULTADO_ERROR;
139:     }
140:     free(line_ptr);
141:     return resultado;
142: }
```

```
    1: #include <sys/regdef.h>
    2: #include <sys/syscall.h>
    3:
    4: #include "hash.h"
    5:
    6:     .abicalls
    7:     .rdata
    8:     .align 0
    9:
   10: mensaje_error_assert: .asciiz "Error en el assert\n"
   11: longitud_msj_error: .word 19
   12:
   13:     .text
   14:     .align 2
   15:     .globl string_hash_more
   16:     .ent string_hash_more
   17:
   18:     .extern STRING_HASH_INIT
   19:     .extern STRING_HASH_MORE
   20:
   21: string_hash_more:
   22:     .frame fp, 8, ra
   23:     .set noreorder
   24:     .cpload t9
   25:     .set reorder
   26:
   27:     subu sp, sp, 8
   28:     .cprestore 0
   29:         sw fp,4(sp)
   30:     move fp,sp
   31:
   32:     #En el ABA de la caller
   33:     sw a0, 8(fp)        #direccion de sh esta en fp+8
   34:         sw a1, 12(fp)   #direccion de str esta en fp+12
   35:         sw a2, 16(fp)   #direccion de len esta en fp+16
   36:
   37:     lw t2, 8(fp)        #t2=direccion de sh
   38:     lb t0, 0(t2)
   39:
   40:     li t1, STRING_HASH_INIT
   41:
   42:     #si sh->flag == STRING_HASH_INIT salto a flag_init
   43:     beq t0, t1, flag_init
   44:
   45:     li t1,STRING_HASH_MORE
   46:     #si sh->flag == STRING_HASH_MORE salto al while
   47:     beq t0, t1, while
   48:
   49: assertion_fail:
   50:     li a0,2
   51:     la a1, mensaje_error_assert
   52:     lw a2, longitud_msj_error
   53:     li v0, SYS_write
   54:     syscall
   55:     li v0, SYS_exit
   56:     syscall
   57:
   58:
   59: flag_init:
   60:     li t0, STRING_HASH_MORE
   61:     #modifico el struct, sh->flag = STRING_HASH_MORE
   62:     sb t0, 0(t2)
   63:
   64:     #recupero *str en t0
   65:     # cargar 12(fp) y hago lb
   66:     lw t2, 12(fp)
   67:     lb t0, 0(t2)
   68:
```

```
   69:     #(*str)<<7
   70:     sll t0, t0, 7
   71:     #Guardo ese valor en sh->hash
   72:     sw t0, 4(a0)
   73:
   74: while:
   75:     #str en t0
   76:     lw t0, 12(fp)
   77:     # len en t1
   78:     lw t1, 16(fp)
   79:
   80:     #recupero en t2 *str (un char)
   81:     lb t2, 0(t0)
   82:     beqz t2, salir
   83:
   84:     #evaluo len:
   85:     beqz t1, salir
   86:     #decremento y actualizo len
   87:     addi t1, t1, -1
   88:     sw t1, 16(fp)
   89:
   90:     #En t3 cargo el primo
   91:     li t3, 1000003
   92:     #En t4 recupero el valor de sh->hash
   93:     lw t4, 4(a0)
   94:     #En t5 multiplico
   95:     mul t5, t4, t3
   96:     #*str ya lo tengo en t2
   97:     #en t6 hago el xor
   98:     xor t6, t5, t2
   99:     #Actualizo el valor de sh->hash
  100:     sw t6, 4(a0)
  101:
  102:     #incremento str (el puntero).
  103:     addiu t0, t0, 1
  104:     sw t0, 12(fp)
  105:
  106:     #En t7 cargo sh->size
  107:     lw t7, 8(a0)
  108:     #incremento y guardo
  109:     addiu t7, t7, 1
  110:     sw t7, 8(a0)
  111:
  112:     j while
  113:
  114: salir:
  115:     lw fp, 4(sp)
  116:     addiu sp,sp,8
  117:     jr ra
  118:
  119:     .end string_hash_more
```

```c
#include "hasheador.h"

#define MSG_ERROR_INIT "Error al inicializar el hasheador en main."
#define MSG_ERROR_CORRER "Error en la ejecucion del hasheador en main."

int main(int argc, const char *argv[]) {

    hasheador_t hasheador;
    int resultado = hasheador_inicializar(&hasheador, argc, argv);
    if (resultado == ESTADO_ERROR) {
        fprintf(stderr, "%s\n", MSG_ERROR_INIT);
        return resultado;
    }

    resultado = hasheador_correr(&hasheador);
    if (resultado == RESULTADO_ERROR) {
        fprintf(stderr, "%s\n", MSG_ERROR_CORRER);
        hasheador_destruir(&hasheador);
        return resultado;
    }
    hasheador_destruir(&hasheador);
    return RESULTADO_OK;
}
```

```
 1: #include <stdio.h>
 2: #include <stdint.h>
 3: #include <string.h>
 4: #include <assert.h>
 5:
 6: #include "hash.h"
 7:
 8: typedef struct {
 9:         int32_t hash;
10:         char *msg;
11: } regression;
12:
13: int32_t
14: get_hash_(string_hash *sh, char *msg, size_t len, size_t stride)
15: {
16:         char *ptr = msg;
17:         size_t delta;
18:         size_t rem;
19:
20:         string_hash_init(sh);
21:         for (rem = len; rem > 0; ) {
22:                 if (rem >= stride)
23:                         delta = stride;
24:                 else
25:                         delta = rem;
26:
27:                 string_hash_more(sh, ptr, delta);
28:                 rem -= delta;
29:                 ptr += delta;
30:         }
31:         string_hash_done(sh);
32:
33:         return string_hash_value(sh);
34: }
35:
36: int
37: get_hash(char *msg)
38: {
39:         size_t len = strlen(msg);
40:         size_t stride;
41:         string_hash sh;
42:         int32_t h0;
43:         int32_t h;
44:
45:         if (len > 1) {
46:                 h0 = get_hash_(&sh, msg, len, len);
47:
48:                 for (stride = len; stride >= 1; stride--) {
49:                         h = get_hash_(&sh, msg, len, stride);
50:                         assert(h0 == h);
51:                 }
52:         }
53:
54:         return h0;
55: }
56:
57: int
58: main(int argc, char * const argv[])
59: {
60:         regression regressions[] = {
61:                 { 0xcc2b6c5a, "66.20 Organizacion de Computadoras\n" },
62:                 { 0xcb5af1f1, "TP 1 - Segundo Cuatrimestre, 2020\n"  },
63:                 { 0xcb5af1f1, "\n"                                   },
64:                 { 0xd788c5a5, "Archivo de prueba TP 1.\n"            },
65:                 { 0x91ff4b5b, "1\n"                                  },
66:                 { -1, NULL                                           },
67:         };
68:         regression *iter;
```

```
69:         int32_t hash;
70:
71:         for (iter = regressions; iter->msg != NULL; iter++) {
72:                 hash = get_hash(iter->msg);
73:                 printf("0x%08x %s", hash, iter->msg);
74:                 assert(iter->hash == hash);
75:         }
76:
77:         return 0;
78: }
```

B. Código fuente MIPS32

A continuación se presenta el código ensamblador generado por `gcc`, luego de correrlo con el flag `-S`.

```
    1:          .file   1 "hash.c"
    2:          .section .mdebug.abi32
    3:          .previous
    4:          .nan    legacy
    5:          .module fp=xx
    6:          .module nooddspreg
    7:          .abicalls
    8:          .text
    9:          .align  2
   10:          .globl  string_hash_init
   11:          .set    nomips16
   12:          .set    nomicromips
   13:          .ent    string_hash_init
   14:          .type   string_hash_init, @function
   15: string_hash_init:
   16:          .frame  $fp,8,$31               # vars= 0, regs= 1/0, args= 0, gp= 0
   17:          .mask   0x40000000,-4
   18:          .fmask  0x00000000,0
   19:          .set    noreorder
   20:          .set    nomacro
   21:          addiu   $sp,$sp,-8
   22:          sw      $fp,4($sp)
   23:          move    $fp,$sp
   24:          sw      $4,8($fp)
   25:          lw      $2,8($fp)
   26:          li      $3,1                    # 0x1
   27:          sb      $3,0($2)
   28:          lw      $2,8($fp)
   29:          sw      $0,4($2)
   30:          lw      $2,8($fp)
   31:          sw      $0,8($2)
   32:          nop
   33:          move    $sp,$fp
   34:          lw      $fp,4($sp)
   35:          addiu   $sp,$sp,8
   36:          jr      $31
   37:          nop
   38:
   39:          .set    macro
   40:          .set    reorder
   41:          .end    string_hash_init
   42:          .size   string_hash_init, .-string_hash_init
   43:          .rdata
   44:          .align  2
   45: $LC0:
   46:          .ascii  "hash.c\000"
   47:          .align  2
   48: $LC1:
   49:          .ascii  "sh->flag == STRING_HASH_INIT || sh->flag == STRING_HASH_"
   50:          .ascii  "MORE\000"
   51:          .text
   52:          .align  2
   53:          .globl  string_hash_done
   54:          .set    nomips16
   55:          .set    nomicromips
   56:          .ent    string_hash_done
   57:          .type   string_hash_done, @function
   58: string_hash_done:
   59:          .frame  $fp,32,$31              # vars= 0, regs= 2/0, args= 16, gp= 8
   60:          .mask   0xc0000000,-4
   61:          .fmask  0x00000000,0
   62:          .set    noreorder
   63:          .cpload $25
   64:          .set    nomacro
   65:          addiu   $sp,$sp,-32
   66:          sw      $31,28($sp)
   67:          sw      $fp,24($sp)
   68:          move    $fp,$sp
```

```
   69:         .cprestore      16
   70:         sw      $4,32($fp)
   71:         lw      $2,32($fp)
   72:         lb      $3,0($2)
   73:         li      $2,1                    # 0x1
   74:         beq     $3,$2,$L3
   75:         nop
   76:
   77:         lw      $2,32($fp)
   78:         lb      $3,0($2)
   79:         li      $2,2                    # 0x2
   80:         beq     $3,$2,$L3
   81:         nop
   82:
   83:         lw      $2,%got(__PRETTY_FUNCTION__.1636)($28)
   84:         addiu   $7,$2,%lo(__PRETTY_FUNCTION__.1636)
   85:         li      $6,17                   # 0x11
   86:         lw      $2,%got($LC0)($28)
   87:         addiu   $5,$2,%lo($LC0)
   88:         lw      $2,%got($LC1)($28)
   89:         addiu   $4,$2,%lo($LC1)
   90:         lw      $2,%call16(__assert_fail)($28)
   91:         move    $25,$2
   92:         .reloc  1f,R_MIPS_JALR,__assert_fail
   93: 1:      jalr    $25
   94:         nop
   95:
   96: $L3:
   97:         lw      $2,32($fp)
   98:         lw      $2,4($2)
   99:         move    $3,$2
  100:         lw      $2,32($fp)
  101:         lw      $2,8($2)
  102:         xor     $2,$3,$2
  103:         move    $3,$2
  104:         lw      $2,32($fp)
  105:         sw      $3,4($2)
  106:         lw      $2,32($fp)
  107:         lw      $3,4($2)
  108:         li      $2,-1                   # 0xffffffffffffffff
  109:         bne     $3,$2,$L4
  110:         nop
  111:
  112:         lw      $2,32($fp)
  113:         li      $3,-2                   # 0xfffffffffffffffe
  114:         sw      $3,4($2)
  115: $L4:
  116:         lw      $2,32($fp)
  117:         li      $3,3                    # 0x3
  118:         sb      $3,0($2)
  119:         nop
  120:         move    $sp,$fp
  121:         lw      $31,28($sp)
  122:         lw      $fp,24($sp)
  123:         addiu   $sp,$sp,32
  124:         jr      $31
  125:         nop
  126:
  127:         .set    macro
  128:         .set    reorder
  129:         .end    string_hash_done
  130:         .size   string_hash_done, .-string_hash_done
  131:         .align  2
  132:         .globl  string_hash_value
  133:         .set    nomips16
  134:         .set    nomicromips
  135:         .ent    string_hash_value
  136:         .type   string_hash_value, @function
```

```
  137: string_hash_value:
  138:          .frame  $fp,8,$31               # vars= 0, regs= 1/0, args= 0, gp= 0
  139:          .mask   0x40000000,-4
  140:          .fmask  0x00000000,0
  141:          .set    noreorder
  142:          .set    nomacro
  143:          addiu   $sp,$sp,-8
  144:          sw      $fp,4($sp)
  145:          move    $fp,$sp
  146:          sw      $4,8($fp)
  147:          lw      $2,8($fp)
  148:          lw      $2,4($2)
  149:          move    $sp,$fp
  150:          lw      $fp,4($sp)
  151:          addiu   $sp,$sp,8
  152:          jr      $31
  153:          nop
  154:
  155:          .set    macro
  156:          .set    reorder
  157:          .end    string_hash_value
  158:          .size   string_hash_value, .-string_hash_value
  159:          .rdata
  160:          .align  2
  161:          .type   __PRETTY_FUNCTION__.1636, @object
  162:          .size   __PRETTY_FUNCTION__.1636, 17
  163: __PRETTY_FUNCTION__.1636:
  164:          .ascii  "string_hash_done\000"
  165:          .ident  "GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516"
```

```
    1:          .file   1 "hasheador.c"
    2:          .section .mdebug.abi32
    3:          .previous
    4:          .nan    legacy
    5:          .module fp=xx
    6:          .module nooddspreg
    7:          .abicalls
    8:          .text
    9:          .align  2
   10:          .set    nomips16
   11:          .set    nomicromips
   12:          .ent    str_en_array
   13:          .type   str_en_array, @function
   14: str_en_array:
   15:          .frame  $fp,40,$31              # vars= 8, regs= 2/0, args= 16, gp= 8
   16:          .mask   0xc0000000,-4
   17:          .fmask  0x00000000,0
   18:          .set    noreorder
   19:          .cpload $25
   20:          .set    nomacro
   21:          addiu   $sp,$sp,-40
   22:          sw      $31,36($sp)
   23:          sw      $fp,32($sp)
   24:          move    $fp,$sp
   25:          .cprestore      16
   26:          sw      $4,40($fp)
   27:          sw      $5,44($fp)
   28:          sw      $6,48($fp)
   29:          sw      $7,52($fp)
   30:          sw      $0,24($fp)
   31:          b       $L2
   32:          nop
   33:
   34: $L6:
   35:          lw      $2,24($fp)
   36:          sll     $2,$2,2
   37:          lw      $3,44($fp)
   38:          addu    $2,$3,$2
   39:          lw      $2,0($2)
   40:          lw      $5,40($fp)
   41:          move    $4,$2
   42:          lw      $2,%call16(strcmp)($28)
   43:          move    $25,$2
   44:          .reloc  1f,R_MIPS_JALR,strcmp
   45: 1:       jalr    $25
   46:          nop
   47:
   48:          lw      $28,16($fp)
   49:          bne     $2,$0,$L3
   50:          nop
   51:
   52:          lw      $2,52($fp)
   53:          beq     $2,$0,$L4
   54:          nop
   55:
   56:          lw      $2,52($fp)
   57:          lw      $3,24($fp)
   58:          sw      $3,0($2)
   59: $L4:
   60:          li      $2,1                    # 0x1
   61:          b       $L5
   62:          nop
   63:
   64: $L3:
   65:          lw      $2,24($fp)
   66:          addiu   $2,$2,1
   67:          sw      $2,24($fp)
   68: $L2:
```

```
   69:         lw      $3,24($fp)
   70:         lw      $2,48($fp)
   71:         sltu    $2,$3,$2
   72:         bne     $2,$0,$L6
   73:         nop
   74:
   75:         move    $2,$0
   76: $L5:
   77:         move    $sp,$fp
   78:         lw      $31,36($sp)
   79:         lw      $fp,32($sp)
   80:         addiu   $sp,$sp,40
   81:         jr      $31
   82:         nop
   83:
   84:         .set    macro
   85:         .set    reorder
   86:         .end    str_en_array
   87:         .size   str_en_array, .-str_en_array
   88:         .rdata
   89:         .align  2
   90: $LC0:
   91:         .ascii  "-h\000"
   92:         .align  2
   93: $LC1:
   94:         .ascii  "-V\000"
   95:         .align  2
   96: $LC2:
   97:         .ascii  "-i\000"
   98:         .align  2
   99: $LC3:
  100:         .ascii  "r\000"
  101:         .align  2
  102: $LC4:
  103:         .ascii  "-o\000"
  104:         .align  2
  105: $LC5:
  106:         .ascii  "w\000"
  107:         .text
  108:         .align  2
  109:         .globl  hasheador_inicializar
  110:         .set    nomips16
  111:         .set    nomicromips
  112:         .ent    hasheador_inicializar
  113:         .type   hasheador_inicializar, @function
  114: hasheador_inicializar:
  115:         .frame  $fp,40,$31              # vars= 8, regs= 2/0, args= 16, gp= 8
  116:         .mask   0xc0000000,-4
  117:         .fmask  0x00000000,0
  118:         .set    noreorder
  119:         .cpload $25
  120:         .set    nomacro
  121:         addiu   $sp,$sp,-40
  122:         sw      $31,36($sp)
  123:         sw      $fp,32($sp)
  124:         move    $fp,$sp
  125:         .cprestore      16
  126:         sw      $4,40($fp)
  127:         sw      $5,44($fp)
  128:         sw      $6,48($fp)
  129:         lw      $2,40($fp)
  130:         li      $3,2                    # 0x2
  131:         sw      $3,12($2)
  132:         lw      $2,40($fp)
  133:         sw      $0,8($2)
  134:         lw      $2,44($fp)
  135:         move    $7,$0
  136:         move    $6,$2
```

```
  137:          lw      $5,48($fp)
  138:          lw      $2,%got($LC0)($28)
  139:          addiu   $4,$2,%lo($LC0)
  140:          lw      $2,%got(str_en_array)($28)
  141:          addiu   $2,$2,%lo(str_en_array)
  142:          move    $25,$2
  143:          .reloc  1f,R_MIPS_JALR,str_en_array
  144: 1:       jalr    $25
  145:          nop
  146:
  147:          lw      $28,16($fp)
  148:          beq     $2,$0,$L8
  149:          nop
  150:
  151:          lw      $2,40($fp)
  152:          sw      $0,12($2)
  153:          b       $L9
  154:          nop
  155:
  156: $L8:
  157:          lw      $2,44($fp)
  158:          move    $7,$0
  159:          move    $6,$2
  160:          lw      $5,48($fp)
  161:          lw      $2,%got($LC1)($28)
  162:          addiu   $4,$2,%lo($LC1)
  163:          lw      $2,%got(str_en_array)($28)
  164:          addiu   $2,$2,%lo(str_en_array)
  165:          move    $25,$2
  166:          .reloc  1f,R_MIPS_JALR,str_en_array
  167: 1:       jalr    $25
  168:          nop
  169:
  170:          lw      $28,16($fp)
  171:          beq     $2,$0,$L9
  172:          nop
  173:
  174:          lw      $2,40($fp)
  175:          li      $3,1                    # 0x1
  176:          sw      $3,12($2)
  177: $L9:
  178:          sw      $0,24($fp)
  179:          lw      $2,44($fp)
  180:          addiu   $3,$fp,24
  181:          move    $7,$3
  182:          move    $6,$2
  183:          lw      $5,48($fp)
  184:          lw      $2,%got($LC2)($28)
  185:          addiu   $4,$2,%lo($LC2)
  186:          lw      $2,%got(str_en_array)($28)
  187:          addiu   $2,$2,%lo(str_en_array)
  188:          move    $25,$2
  189:          .reloc  1f,R_MIPS_JALR,str_en_array
  190: 1:       jalr    $25
  191:          nop
  192:
  193:          lw      $28,16($fp)
  194:          beq     $2,$0,$L10
  195:          nop
  196:
  197:          lw      $2,24($fp)
  198:          addiu   $2,$2,1
  199:          sll     $2,$2,2
  200:          lw      $3,48($fp)
  201:          addu    $2,$3,$2
  202:          lw      $2,0($2)
  203:          lb      $3,0($2)
  204:          li      $2,45                   # 0x2d
```

```
  205:          bne     $3,$2,$L11
  206:          nop
  207:
  208:          lw      $2,%got(stdin)($28)
  209:          lw      $3,0($2)
  210:          lw      $2,40($fp)
  211:          sw      $3,0($2)
  212:          b       $L12
  213:          nop
  214:
  215: $L11:
  216:          lw      $2,24($fp)
  217:          addiu   $2,$2,1
  218:          sll     $2,$2,2
  219:          lw      $3,48($fp)
  220:          addu    $2,$3,$2
  221:          lw      $3,0($2)
  222:          lw      $2,%got($LC3)($28)
  223:          addiu   $5,$2,%lo($LC3)
  224:          move    $4,$3
  225:          lw      $2,%call16(fopen)($28)
  226:          move    $25,$2
  227:          .reloc  1f,R_MIPS_JALR,fopen
  228: 1:       jalr    $25
  229:          nop
  230:
  231:          lw      $28,16($fp)
  232:          move    $3,$2
  233:          lw      $2,40($fp)
  234:          sw      $3,0($2)
  235:          lw      $2,40($fp)
  236:          lw      $2,0($2)
  237:          bne     $2,$0,$L12
  238:          nop
  239:
  240:          lw      $2,40($fp)
  241:          li      $3,-1                   # 0xffffffffffffffff
  242:          sw      $3,8($2)
  243:          b       $L13
  244:          nop
  245:
  246: $L10:
  247:          lw      $2,%got(stdin)($28)
  248:          lw      $3,0($2)
  249:          lw      $2,40($fp)
  250:          sw      $3,0($2)
  251: $L12:
  252:          lw      $2,44($fp)
  253:          addiu   $3,$fp,24
  254:          move    $7,$3
  255:          move    $6,$2
  256:          lw      $5,48($fp)
  257:          lw      $2,%got($LC4)($28)
  258:          addiu   $4,$2,%lo($LC4)
  259:          lw      $2,%got(str_en_array)($28)
  260:          addiu   $2,$2,%lo(str_en_array)
  261:          move    $25,$2
  262:          .reloc  1f,R_MIPS_JALR,str_en_array
  263: 1:       jalr    $25
  264:          nop
  265:
  266:          lw      $28,16($fp)
  267:          beq     $2,$0,$L14
  268:          nop
  269:
  270:          lw      $2,24($fp)
  271:          addiu   $2,$2,1
  272:          sll     $2,$2,2
```

```
  273:          lw      $3,48($fp)
  274:          addu    $2,$3,$2
  275:          lw      $2,0($2)
  276:          lb      $3,0($2)
  277:          li      $2,45                   # 0x2d
  278:          bne     $3,$2,$L15
  279:          nop
  280:
  281:          lw      $2,%got(stdout)($28)
  282:          lw      $3,0($2)
  283:          lw      $2,40($fp)
  284:          sw      $3,4($2)
  285:          b       $L13
  286:          nop
  287:
  288: $L15:
  289:          lw      $2,24($fp)
  290:          addiu   $2,$2,1
  291:          sll     $2,$2,2
  292:          lw      $3,48($fp)
  293:          addu    $2,$3,$2
  294:          lw      $3,0($2)
  295:          lw      $2,%got($LC5)($28)
  296:          addiu   $5,$2,%lo($LC5)
  297:          move    $4,$3
  298:          lw      $2,%call16(fopen)($28)
  299:          move    $25,$2
  300:          .reloc  1f,R_MIPS_JALR,fopen
  301: 1:       jalr    $25
  302:          nop
  303:
  304:          lw      $28,16($fp)
  305:          move    $3,$2
  306:          lw      $2,40($fp)
  307:          sw      $3,4($2)
  308:          lw      $2,40($fp)
  309:          lw      $2,4($2)
  310:          bne     $2,$0,$L13
  311:          nop
  312:
  313:          lw      $2,40($fp)
  314:          li      $3,-1                   # 0xffffffffffffffff
  315:          sw      $3,8($2)
  316:          lw      $2,40($fp)
  317:          lw      $3,0($2)
  318:          lw      $2,%got(stdin)($28)
  319:          lw      $2,0($2)
  320:          beq     $3,$2,$L13
  321:          nop
  322:
  323:          lw      $2,40($fp)
  324:          lw      $2,0($2)
  325:          move    $4,$2
  326:          lw      $2,%call16(fclose)($28)
  327:          move    $25,$2
  328:          .reloc  1f,R_MIPS_JALR,fclose
  329: 1:       jalr    $25
  330:          nop
  331:
  332:          lw      $28,16($fp)
  333:          b       $L13
  334:          nop
  335:
  336: $L14:
  337:          lw      $2,%got(stdout)($28)
  338:          lw      $3,0($2)
  339:          lw      $2,40($fp)
  340:          sw      $3,4($2)
```

```
341: $L13:
342:         lw      $2,40($fp)
343:         lw      $2,8($2)
344:         move    $sp,$fp
345:         lw      $31,36($sp)
346:         lw      $fp,32($sp)
347:         addiu   $sp,$sp,40
348:         jr      $31
349:         nop
350:
351:         .set    macro
352:         .set    reorder
353:         .end    hasheador_inicializar
354:         .size   hasheador_inicializar, .-hasheador_inicializar
355:         .align  2
356:         .globl  hasheador_destruir
357:         .set    nomips16
358:         .set    nomicromips
359:         .ent    hasheador_destruir
360:         .type   hasheador_destruir, @function
361: hasheador_destruir:
362:         .frame  $fp,32,$31              # vars= 0, regs= 2/0, args= 16, gp= 8
363:         .mask   0xc0000000,-4
364:         .fmask  0x00000000,0
365:         .set    noreorder
366:         .cpload $25
367:         .set    nomacro
368:         addiu   $sp,$sp,-32
369:         sw      $31,28($sp)
370:         sw      $fp,24($sp)
371:         move    $fp,$sp
372:         .cprestore      16
373:         sw      $4,32($fp)
374:         lw      $2,32($fp)
375:         lw      $3,8($2)
376:         li      $2,-1                   # 0xffffffffffffffff
377:         beq     $3,$2,$L22
378:         nop
379:
380:         lw      $2,32($fp)
381:         lw      $3,0($2)
382:         lw      $2,%got(stdin)($28)
383:         lw      $2,0($2)
384:         beq     $3,$2,$L21
385:         nop
386:
387:         lw      $2,%got(stdin)($28)
388:         lw      $2,0($2)
389:         move    $4,$2
390:         lw      $2,%call16(fclose)($28)
391:         move    $25,$2
392:         .reloc  1f,R_MIPS_JALR,fclose
393: 1:      jalr    $25
394:         nop
395:
396:         lw      $28,16($fp)
397: $L21:
398:         lw      $2,32($fp)
399:         lw      $3,4($2)
400:         lw      $2,%got(stdout)($28)
401:         lw      $2,0($2)
402:         beq     $3,$2,$L18
403:         nop
404:
405:         lw      $2,%got(stdout)($28)
406:         lw      $2,0($2)
407:         move    $4,$2
408:         lw      $2,%call16(fclose)($28)
```

```
409:         move    $25,$2
410:         .reloc  1f,R_MIPS_JALR,fclose
411: 1:      jalr    $25
412:         nop
413:
414:         lw      $28,16($fp)
415:         b       $L18
416:         nop
417:
418: $L22:
419:         nop
420: $L18:
421:         move    $sp,$fp
422:         lw      $31,28($sp)
423:         lw      $fp,24($sp)
424:         addiu   $sp,$sp,32
425:         jr      $31
426:         nop
427:
428:         .set    macro
429:         .set    reorder
430:         .end    hasheador_destruir
431:         .size   hasheador_destruir, .-hasheador_destruir
432:         .rdata
433:         .align  2
434: $LC6:
435:         .ascii  "Usage:\012tp1 -h\012tp1 -V\012tp1 -i in_file -o out_file"
436:         .ascii  "\012Options:\012-V, --version\012Print version and quit."
437:         .ascii  "\012-h, --help\012Print this information and quit.\012-i"
438:         .ascii  ", --input\012Specify input stream/file, \"-\" for stdin."
439:         .ascii  "\012-o, --output\012Specify output stream/file, \"-\" fo"
440:         .ascii  "r stdout.\012Examples:\012tp1 < in.txt > out.txt\012cat "
441:         .ascii  "in.txt | tp1 -i - > out.txt\000"
442:         .align  2
443: $LC7:
444:         .ascii  "version 0.1\000"
445:         .text
446:         .align  2
447:         .globl  hasheador_correr
448:         .set    nomips16
449:         .set    nomicromips
450:         .ent    hasheador_correr
451:         .type   hasheador_correr, @function
452: hasheador_correr:
453:         .frame  $fp,40,$31              # vars= 8, regs= 2/0, args= 16, gp= 8
454:         .mask   0xc0000000,-4
455:         .fmask  0x00000000,0
456:         .set    noreorder
457:         .cpload $25
458:         .set    nomacro
459:         addiu   $sp,$sp,-40
460:         sw      $31,36($sp)
461:         sw      $fp,32($sp)
462:         move    $fp,$sp
463:         .cprestore      16
464:         sw      $4,40($fp)
465:         sw      $0,24($fp)
466:         lw      $2,40($fp)
467:         lw      $3,8($2)
468:         li      $2,-1                   # 0xffffffffffffffff
469:         bne     $3,$2,$L24
470:         nop
471:
472:         lw      $2,%got($LC6)($28)
473:         addiu   $4,$2,%lo($LC6)
474:         lw      $2,%call16(puts)($28)
475:         move    $25,$2
476:         .reloc  1f,R_MIPS_JALR,puts
```

```
477: 1:      jalr    $25
478:         nop
479:
480:         lw      $28,16($fp)
481:         li      $2,-1                   # 0xffffffffffffffff
482:         sw      $2,24($fp)
483:         b       $L25
484:         nop
485:
486: $L24:
487:         lw      $2,40($fp)
488:         lw      $2,12($2)
489:         bne     $2,$0,$L26
490:         nop
491:
492:         lw      $2,%got($LC6)($28)
493:         addiu   $4,$2,%lo($LC6)
494:         lw      $2,%call16(puts)($28)
495:         move    $25,$2
496:         .reloc  1f,R_MIPS_JALR,puts
497: 1:      jalr    $25
498:         nop
499:
500:         lw      $28,16($fp)
501:         sw      $0,24($fp)
502:         b       $L25
503:         nop
504:
505: $L26:
506:         lw      $2,40($fp)
507:         lw      $3,12($2)
508:         li      $2,1                    # 0x1
509:         bne     $3,$2,$L27
510:         nop
511:
512:         lw      $2,%got($LC7)($28)
513:         addiu   $4,$2,%lo($LC7)
514:         lw      $2,%call16(puts)($28)
515:         move    $25,$2
516:         .reloc  1f,R_MIPS_JALR,puts
517: 1:      jalr    $25
518:         nop
519:
520:         lw      $28,16($fp)
521:         sw      $0,24($fp)
522:         b       $L25
523:         nop
524:
525: $L27:
526:         lw      $4,40($fp)
527:         lw      $2,%got(hasheador_hashear_archivo)($28)
528:         move    $25,$2
529:         .reloc  1f,R_MIPS_JALR,hasheador_hashear_archivo
530: 1:      jalr    $25
531:         nop
532:
533:         lw      $28,16($fp)
534:         sw      $2,24($fp)
535: $L25:
536:         lw      $2,24($fp)
537:         move    $sp,$fp
538:         lw      $31,36($sp)
539:         lw      $fp,32($sp)
540:         addiu   $sp,$sp,40
541:         jr      $31
542:         nop
543:
544:         .set    macro
```

```
  545:          .set    reorder
  546:          .end    hasheador_correr
  547:          .size   hasheador_correr, .-hasheador_correr
  548:          .rdata
  549:          .align  2
  550: $LC8:
  551:          .ascii  "0x%08x %s\000"
  552:          .text
  553:          .align  2
  554:          .globl  hasheador_hashear_archivo
  555:          .set    nomips16
  556:          .set    nomicromips
  557:          .ent    hasheador_hashear_archivo
  558:          .type   hasheador_hashear_archivo, @function
  559: hasheador_hashear_archivo:
  560:          .frame  $fp,64,$31              # vars= 24, regs= 3/0, args= 16, gp= 8
  561:          .mask   0xc0010000,-4
  562:          .fmask  0x00000000,0
  563:          .set    noreorder
  564:          .cpload $25
  565:          .set    nomacro
  566:          addiu   $sp,$sp,-64
  567:          sw      $31,60($sp)
  568:          sw      $fp,56($sp)
  569:          sw      $16,52($sp)
  570:          move    $fp,$sp
  571:          .cprestore      16
  572:          sw      $4,64($fp)
  573:          sw      $0,40($fp)
  574:          sw      $0,44($fp)
  575:          sw      $0,24($fp)
  576:          b       $L30
  577:          nop
  578:
  579: $L31:
  580:          addiu   $2,$fp,28
  581:          move    $4,$2
  582:          lw      $2,%call16(string_hash_init)($28)
  583:          move    $25,$2
  584:          .reloc  1f,R_MIPS_JALR,string_hash_init
  585: 1:       jalr    $25
  586:          nop
  587:
  588:          lw      $28,16($fp)
  589:          lw      $3,40($fp)
  590:          lw      $4,44($fp)
  591:          addiu   $2,$fp,28
  592:          move    $6,$4
  593:          move    $5,$3
  594:          move    $4,$2
  595:          lw      $2,%call16(string_hash_more)($28)
  596:          move    $25,$2
  597:          .reloc  1f,R_MIPS_JALR,string_hash_more
  598: 1:       jalr    $25
  599:          nop
  600:
  601:          lw      $28,16($fp)
  602:          addiu   $2,$fp,28
  603:          move    $4,$2
  604:          lw      $2,%call16(string_hash_done)($28)
  605:          move    $25,$2
  606:          .reloc  1f,R_MIPS_JALR,string_hash_done
  607: 1:       jalr    $25
  608:          nop
  609:
  610:          lw      $28,16($fp)
  611:          lw      $2,64($fp)
  612:          lw      $16,4($2)
```

```
  613:          addiu   $2,$fp,28
  614:          move    $4,$2
  615:          lw      $2,%call16(string_hash_value)($28)
  616:          move    $25,$2
  617:          .reloc  1f,R_MIPS_JALR,string_hash_value
  618: 1:       jalr    $25
  619:          nop
  620:
  621:          lw      $28,16($fp)
  622:          move    $3,$2
  623:          lw      $2,40($fp)
  624:          move    $7,$2
  625:          move    $6,$3
  626:          lw      $2,%got($LC8)($28)
  627:          addiu   $5,$2,%lo($LC8)
  628:          move    $4,$16
  629:          lw      $2,%call16(fprintf)($28)
  630:          move    $25,$2
  631:          .reloc  1f,R_MIPS_JALR,fprintf
  632: 1:       jalr    $25
  633:          nop
  634:
  635:          lw      $28,16($fp)
  636: $L30:
  637:          lw      $2,64($fp)
  638:          lw      $4,0($2)
  639:          addiu   $3,$fp,44
  640:          addiu   $2,$fp,40
  641:          move    $6,$4
  642:          move    $5,$3
  643:          move    $4,$2
  644:          lw      $2,%call16(getline)($28)
  645:          move    $25,$2
  646:          .reloc  1f,R_MIPS_JALR,getline
  647: 1:       jalr    $25
  648:          nop
  649:
  650:          lw      $28,16($fp)
  651:          sw      $2,24($fp)
  652:          lw      $3,24($fp)
  653:          li      $2,-1                   # 0xffffffffffffffff
  654:          bne     $3,$2,$L31
  655:          nop
  656:
  657:          lw      $2,64($fp)
  658:          lw      $2,0($2)
  659:          move    $4,$2
  660:          lw      $2,%call16(feof)($28)
  661:          move    $25,$2
  662:          .reloc  1f,R_MIPS_JALR,feof
  663: 1:       jalr    $25
  664:          nop
  665:
  666:          lw      $28,16($fp)
  667:          beq     $2,$0,$L32
  668:          nop
  669:
  670:          sw      $0,24($fp)
  671:          b       $L33
  672:          nop
  673:
  674: $L32:
  675:          li      $2,-1                   # 0xffffffffffffffff
  676:          sw      $2,24($fp)
  677: $L33:
  678:          lw      $2,40($fp)
  679:          move    $4,$2
  680:          lw      $2,%call16(free)($28)
```

```
681:          move    $25,$2
682:          .reloc  1f,R_MIPS_JALR,free
683: 1:       jalr    $25
684:          nop
685:
686:          lw      $28,16($fp)
687:          lw      $2,24($fp)
688:          move    $sp,$fp
689:          lw      $31,60($sp)
690:          lw      $fp,56($sp)
691:          lw      $16,52($sp)
692:          addiu   $sp,$sp,64
693:          jr      $31
694:          nop
695:
696:          .set    macro
697:          .set    reorder
698:          .end    hasheador_hashear_archivo
699:          .size   hasheador_hashear_archivo, .-hasheador_hashear_archivo
700:          .ident  "GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516"
```

```
    1:          .file   1 "tp1.c"
    2:          .section .mdebug.abi32
    3:          .previous
    4:          .nan    legacy
    5:          .module fp=xx
    6:          .module nooddspreg
    7:          .abicalls
    8:          .rdata
    9:          .align  2
   10: $LC0:
   11:          .ascii  "Error al inicializar el hasheador en main.\000"
   12:          .align  2
   13: $LC1:
   14:          .ascii  "%s\012\000"
   15:          .align  2
   16: $LC2:
   17:          .ascii  "Error en la ejecucion del hasheador en main.\000"
   18:          .text
   19:          .align  2
   20:          .globl  main
   21:          .set    nomips16
   22:          .set    nomicromips
   23:          .ent    main
   24:          .type   main, @function
   25: main:
   26:          .frame  $fp,56,$31              # vars= 24, regs= 2/0, args= 16, gp= 8
   27:          .mask   0xc0000000,-4
   28:          .fmask  0x00000000,0
   29:          .set    noreorder
   30:          .cpload $25
   31:          .set    nomacro
   32:          addiu   $sp,$sp,-56
   33:          sw      $31,52($sp)
   34:          sw      $fp,48($sp)
   35:          move    $fp,$sp
   36:          .cprestore      16
   37:          sw      $4,56($fp)
   38:          sw      $5,60($fp)
   39:          addiu   $2,$fp,28
   40:          lw      $6,60($fp)
   41:          lw      $5,56($fp)
   42:          move    $4,$2
   43:          lw      $2,%call16(hasheador_inicializar)($28)
   44:          move    $25,$2
   45:          .reloc  1f,R_MIPS_JALR,hasheador_inicializar
   46: 1:       jalr    $25
   47:          nop
   48:
   49:          lw      $28,16($fp)
   50:          sw      $2,24($fp)
   51:          lw      $3,24($fp)
   52:          li      $2,-1                   # 0xffffffffffffffff
   53:          bne     $3,$2,$L2
   54:          nop
   55:
   56:          lw      $2,%got(stderr)($28)
   57:          lw      $3,0($2)
   58:          lw      $2,%got($LC0)($28)
   59:          addiu   $6,$2,%lo($LC0)
   60:          lw      $2,%got($LC1)($28)
   61:          addiu   $5,$2,%lo($LC1)
   62:          move    $4,$3
   63:          lw      $2,%call16(fprintf)($28)
   64:          move    $25,$2
   65:          .reloc  1f,R_MIPS_JALR,fprintf
   66: 1:       jalr    $25
   67:          nop
   68:
```

```
   69:          lw      $28,16($fp)
   70:          lw      $2,24($fp)
   71:          b       $L5
   72:          nop
   73:
   74: $L2:
   75:          addiu   $2,$fp,28
   76:          move    $4,$2
   77:          lw      $2,%call16(hasheador_correr)($28)
   78:          move    $25,$2
   79:          .reloc  1f,R_MIPS_JALR,hasheador_correr
   80: 1:       jalr    $25
   81:          nop
   82:
   83:          lw      $28,16($fp)
   84:          sw      $2,24($fp)
   85:          lw      $3,24($fp)
   86:          li      $2,-1                   # 0xffffffffffffffff
   87:          bne     $3,$2,$L4
   88:          nop
   89:
   90:          lw      $2,%got(stderr)($28)
   91:          lw      $3,0($2)
   92:          lw      $2,%got($LC2)($28)
   93:          addiu   $6,$2,%lo($LC2)
   94:          lw      $2,%got($LC1)($28)
   95:          addiu   $5,$2,%lo($LC1)
   96:          move    $4,$3
   97:          lw      $2,%call16(fprintf)($28)
   98:          move    $25,$2
   99:          .reloc  1f,R_MIPS_JALR,fprintf
  100: 1:       jalr    $25
  101:          nop
  102:
  103:          lw      $28,16($fp)
  104:          addiu   $2,$fp,28
  105:          move    $4,$2
  106:          lw      $2,%call16(hasheador_destruir)($28)
  107:          move    $25,$2
  108:          .reloc  1f,R_MIPS_JALR,hasheador_destruir
  109: 1:       jalr    $25
  110:          nop
  111:
  112:          lw      $28,16($fp)
  113:          lw      $2,24($fp)
  114:          b       $L5
  115:          nop
  116:
  117: $L4:
  118:          addiu   $2,$fp,28
  119:          move    $4,$2
  120:          lw      $2,%call16(hasheador_destruir)($28)
  121:          move    $25,$2
  122:          .reloc  1f,R_MIPS_JALR,hasheador_destruir
  123: 1:       jalr    $25
  124:          nop
  125:
  126:          lw      $28,16($fp)
  127:          move    $2,$0
  128: $L5:
  129:          move    $sp,$fp
  130:          lw      $31,52($sp)
  131:          lw      $fp,48($sp)
  132:          addiu   $sp,$sp,56
  133:          jr      $31
  134:          nop
  135:
  136:          .set    macro
```

```
  137:          .set    reorder
  138:          .end    main
  139:          .size   main, .-main
  140:          .ident  "GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516"
```

C. Enunciado

El enunciado correspondiente al presente trabajo práctico puede encontrarse en este apéndice.

Universidad de Buenos Aires - FIUBA
66.20 Organización de Computadoras
Trabajo práctico 1: Programación MIPS
2^o cuatrimestre de 2020

$Date: 2020/10/27 23:05:14 $

1. Objetivos

Familiarizarse con el conjunto de instrucciones MIPS y el concepto de ABI, extendiendo un programa que resuelva el problema descripto a continuación.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe impreso de acuerdo con lo que mencionaremos en la sección 6, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Descripción

El programa a desarrollar deberá procesar un *stream* de texto compuesto por una cantidad arbitraria de líneas de longitud arbitraria. A medida que el programa avance en la lectura de cada línea, deberá calcular el hash del contenido de la misma, e imprimir en la salida el valor del hash seguido del contenido de la línea.

Por ejemplo, dado el siguiente flujo de entrada:

```
$ cat input.txt
66.20 Organizacion de Computadoras
TP 1 - Segundo Cuatrimestre, 2020

Archivo de prueba TP 1.
```

Al ejecutar el programa la salida será:

```
$ tp1 -i input.txt -o -
0xcc2b6c5a 66.20 Organizacion de Computadoras
0xcb5af1f1 TP 1 - Segundo Cuatrimestre, 2020
0x4c4b4f0b
0xd788c5a5 Archivo de prueba TP 1.
```

4.1. Ejemplos

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ tp1 -h
Usage:
  tp1 -h
  tp1 -V
  tp1 -i in_file -o out_file
Options:
  -V, --version     Print version and quit.
  -h, --help        Print this information and quit.
  -i, --input       Specify input stream/file, "-" for stdin.
  -o, --output      Specify output stream/file, "-" for stdout.
Examples:
   tp1 < in.txt > out.txt
   cat in.txt | tp1 -i - > out.txt
```

A continuación, ejecutamos algunas pruebas: primero, veamos qué sucede cuando el archivo de entrada está vacío,

```
$ ./tp1 -o salida.txt </dev/null
$ ls -l salida.txt
-rw-r--r-- 1 leandro leandro 0 Oct 20 20:14 salida.txt
```

Aquí puede verse que el programa se comporta según lo esperado, ya que cuando la entrada está vacía, la salida lo estará también.

Veamos qué ocurre al ingresar un archivo con una única línea, la cual contiene un sólo caracter:

```
$ echo 1 | ./tp1 -o -
0x91ff4b5b 1
```

Lo mismo debería ocurrir si la entrada se encuentra alojada en el sistema de archivos:

```
$ echo 1 >entrada.txt
$ ./tp1 -i entrada.txt -o -
0x91ff4b5b 1
```

5. Implementación

El programa a desarrollar constará de una mezcla entre código MIPS32 y C, siendo la parte escrita en *assembly* la encargada de calcular el hash de un bloque de bytes pasado por parámetro. El formato de dicha función será:

```
void string_hash_more(string_hash *, char *, size_t);
```

En donde `string_hash` es un tipo de datos usado para mantener el contexto de la operación de cálculo del hash (opaco para el usuario). En el archivo `hash.c` puede encontrarse una implementación de referencia en lenguaje C de esta función de hash.

6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa;
- Comando(s) para compilar el programa;
- Las corridas de prueba, con los comentarios pertinentes;
- El código fuente, en lenguaje C y MIPS;
- El código MIPS32 generado por el compilador[1];
- Este enunciado.

7. Regresiones

El programa deberá pasar todas las regresiones definidas en el código fuente suministrado en este TP [1]:

```
$ make
cc -Wall -g -o regressions regressions.c hash.c hash.S
:
$ make test
./regressions
0xcc2b6c5a 66.20 Organizacion de Computadoras
0xcb5af1f1 TP 1 - Segundo Cuatrimestre, 2020
0xcb5af1f1
0xd788c5a5 Archivo de prueba TP 1.
0x91ff4b5b 1
```

Asimismo deberá usarse el modo 1 del sistema operativo para manejo de acceso no alineado a memoria [2].

8. Entrega de TPs

La entrega de este trabajo deberá realizarse usando el campus virtual de la materia [3]. Asimismo, en todos los casos, estas presentaciones deberán ser realizadas durante los días martes. El *feedback* estará disponible de un martes hacia el otro, como ocurre durante la modalidad presencial de cursada.

Por otro lado, la última fecha de entrega y presentación para esta trabajo será el martes 10/11.

[1] Por motivos prácticos, en la copia impresa sólo es necesario incluir la primera página del código assembly MIPS32 generado por el compilador.

Referencias

[1] Código fuente para realizar el Trabajo Práctico.
`https://drive.google.com/file/d/1Jdz3b0AoBwipKAAsjP7qsDqR-hSSM0ik/view`

[2] Controlling the kernel unalignment handling via debugfs,
https://www.linux-mips.org/wiki/Alignment.

[3] Aula Virtual - Organización de Computadoras 86.37/66.20 - Curso 1 - Turno Martes.
`https://campus.fi.uba.ar/course/view.php?id=649`